Reuse method for quantum circuit synthesis

Cyril Allouche, Marc Baboulin, Timothée Goubault de Brugière, Benoît Valiron

Reuse method for quantum circuit synthesis

C. Allouche, M. Baboulin, T. Goubault de Brugière and B. Valiron

Abstract The algebraic decomposition of a unitary operator is a key operation in the synthesis of quantum circuits. If most methods factorize the matrix into products, there exists a method that allows to reuse already existing optimized circuits to implement linear combinations of them. This paper presents an attempt to extend this method to a general framework of circuit synthesis. The method needs to find suitable groups for the implementation of new quantum circuits. We identify key points necessary for the construction of a comprehensive method and we test potential group candidates.

1 Introduction

The notion of *quantum circuit* has emerged from the beginning of the field of quantum computing [3] and so far remains the most widespread description of a quantum algorithm. Contrary to conventional algorithms that manipulate bits (0 or 1) using boolean gates, a *quantum algorithm* operates on quantum bits, or *qubits*, using a series of quantum gates which are generally desired as simple as possible. A quantum

C. Allouche
Atos-Bull, Les Clayes-sous-Bois, France
e-mail: cyril.allouche@atos.net

M. Baboulin
LRI and Université Paris-Sud, Orsay, France
e-mail: marc.baboulin@lri.fr

T. Goubault de Brugière
Atos/Bull, LRI and Université Paris-Sud
e-mail: timothee.goubault@lri.fr

B. Valiron
LRI and CentraleSupélec, Orsay, France
e-mail: benoit.valiron@lri.fr

$$\underset{X}{\begin{pmatrix}0&1\\1&0\end{pmatrix}} \quad \underset{Y}{\begin{pmatrix}0&-i\\i&0\end{pmatrix}} \quad \underset{Z}{\begin{pmatrix}1&0\\0&-1\end{pmatrix}} \quad \underset{\text{CNOT}}{\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}} \quad \underset{\text{SWAP}}{\begin{pmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{pmatrix}}$$

$$\underset{H}{\begin{pmatrix}1&1\\1&-1\end{pmatrix}} \quad \underset{T}{\begin{pmatrix}1&0\\0&e^{i\pi/4}\end{pmatrix}}$$

Table 1 Usual elementary unitary matrices.

bit is formally a unit vector in $\mathbb{C}^2$ (modulo a phase factor) and represents a linear superposition of both states 0 and 1. Using the usual Dirac notation, the state $|\psi\rangle$ of one qubit is the vector

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \begin{pmatrix}\alpha\\\beta\end{pmatrix}. \tag{1}$$

We compose spaces of states for systems of several qubits by using the tensor product of the spaces of states of each qubit. Then a system of n qubits is a unit vector that belongs to $\mathbb{C}^{2^n}$. With this formalism, quantum gates are unitary matrices, i.e matrices whose inverse are their own adjoint. Depending on the physical realization of the quantum memory, some unitary matrices might be easier to implement than others [11]: we refer to these gates as *elementary*. Among the elementary quantum gates usually considered, we can mention the gates presented in Table 1: the Pauli matrices X, Y and Z, the Hadamard gate H, the T-gate and the two-qubit gates CNOT and SWAP.

A quantum circuit is then a series of elementary quantum gates operating on n qubits for some $n > 0$. It represents a global quantum operator that corresponds to a matrix of $\mathscr{U}(2^n)$, where $\mathscr{U}(2^n)$ denotes the set of unitary matrices of size $2^n \times 2^n$ (see, e.g., [10] for a comprehensive introduction to quantum computing).

A quantum circuit can be represented as in Figure 1. Each wire corresponds to a quantum bit and we read from left to right the gates that are applied to the system. In this case, we first apply a Hadamard gate on the first qubit (tensored with the identity on the second qubit), then the Pauli gate X is applied to the second qubit, controlled by the first one. This means that if the first qubit was in state $|0\rangle$, the state is unchanged, and if it was in state $|1\rangle$ the gate X is applied. One can check that the controlled-X gate is equivalent to the CNOT gate. Finally, the overall operator U applied to the system is the product

$$U = \Lambda(X) \times (H \otimes I_2), \tag{2}$$

where $\Lambda(X)$ denotes the fact that the gate X is controlled by the first qubit.

A set of gates is said to be *universal* when any unitary can be implemented via a quantum circuit using these gates. Since the mid-1990s various universality results have been shown (see, e.g. [5]). For example the set composed of all the 1-qubit gates and the CNOT gate is sufficient to implement any operator. Another example is the set of H, T and CNOT gates which is also universal. In order to implement a

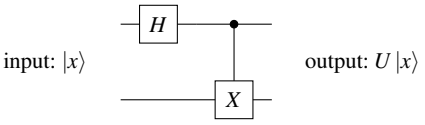

Fig. 1 Example of a quantum circuit

general quantum operator on a concrete system, it is necessary to decompose it into elementary gates. If these elementary gates are chosen from a universal set, then it is theoretically possible to implement this operator.

Quantum computing yields several challenges. One of the problem is to actually generate a quantum circuit from a textual description. Several programming languages have been developed to address this issue [15, 6, 17]. Another problem is to optimize the generated quantum circuits by simplifying them as much as possible, for example by using rewrite rules in order to minimize the number of elementary gates [9]. Also many efforts have been made to provide software that simulates quantum circuits on classical computers in order to help researchers to make progress in view of a future quantum computer. In this case the optimization of circuits can be understood as minimizing the simulation time.

In this paper we are instead interested in the *synthesis of quantum circuits*. Contrary to the case where the circuit is explicitly described, here a unitary operator is provided as a matrix and the problem consists in finding a quantum circuit that implements it optimally.

We can impose various constraints on the solution circuit such as the choice of the considered elementary gates, the physical medium, the arrangement of qubits, the memory properties, etc. We can evaluate the optimality of the solution by measuring

- the number of elementary gates,
- the time to find the circuit,
- the time to classically simulate the circuit,
- the error between the targeted operator and the implemented operator (for example using the norm of the difference between the corresponding matrices).

Over the years more and more efficient methods have been developed to synthesize an arbitrary quantum operator [1, 8, 13, 16]. Most synthesis frameworks rely on linear algebra methods to decompose unitary matrices. The first methods aimed at decomposing the operator column by column [1, 8]. One can cite as example the QR method, via Givens rotations [16]. Other decomposition methods have also been proposed, for example the recent block-ZXZ decomposition [4], or the quantum Shannon decomposition [13] that relies on the use of the sine cosine decomposition of a unitary operator $U \in U(2^n)$:

$$U = \begin{pmatrix} A_1 & 0 \\ 0 & A_2 \end{pmatrix} \begin{pmatrix} C & -S \\ S & C \end{pmatrix} \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix} \tag{3}$$

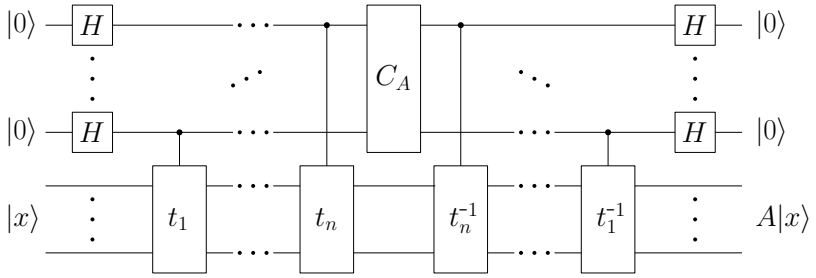

Fig. 2 Quantum circuit implementing a linear combination of operators

where $A_1, A_2, B_1, B_2 \in U(2^{n-1})$ and C, S are diagonal real matrices such that $C^2 + S^2 = I_{2^{n-1}}$. For an overview of the history and links between these various methods, refer to [12].

In this context, there exists a less typical method that focuses on a decomposition of the operator as a linear combination of other operators chosen from a given set. This method enables to reuse optimized circuits in order to implement more complex operators [7]. To our knowledge, this is the only method using such a technique. This method, which we informally call the *reuse method*, has been shown to be efficient on specific cases [7]. Our objective in this paper is to determine whether this method can be efficiently extended to a general framework for circuit synthesis.

The paper is organized as follows. In Section 2, we recall the main principles of the reuse method. In Section 3, we select the groups that can be used in synthesizing circuits via the reuse method. In Section 4, we study the potential group candidates. We conclude in Section 5.

2 The reuse method

The reuse method has emerged from the following motivation: if we know how to implement circuits (supposedly efficiently), can we directly reuse these circuits in order to implement new operators?

Based on this idea, Klappenecker and Rötteler replied in the affirmative [7]. Below is a simplified version of [7, Th. 6].

Theorem *Let $G \subset \mathscr{U}(2^m)$ be a group of order 2^n, and $T = (t_1, ..., t_n)$ be a transversal of G (i.e any member g of G can be written as $g = t_1^{\alpha_1} ... t_n^{\alpha_n}$ with $\alpha_i \in \{0,1\}$). Suppose*

$$A = \sum_{g \in G} \beta_g g \tag{4}$$

with $A \in \mathscr{U}(2^m)$ and define the coefficient matrix $C_A = (\beta_{g^{-1}h})_{g,h \in G}$. Then the coefficients $(\beta_g)_{g \in G}$ can be chosen such that C_A is unitary and the operator A can be implemented as depicted in Figure 2.

This remains a simplified version, sufficient for the rest of our study. An illustration of the method can be found in [7, Sec. 3], where there is an implementation of the Hartley transform via a linear combination of powers of the Fourier transform.

A key point in the use of this method is the distribution of information between the group and the matrix of coefficients. When the group contains sufficient information, such as the Fourier transform powers group, the coefficient matrix is easy to compute and the efficiency of the quantum Fourier transform synthesis is used to produce an efficient circuit on non trivial operators. An alternative would be to consider the problem in the other direction: the group is simple, contains little information but the matrix of coefficients — which now has a maximum of information — has a structure that makes its implementation effective.

For example, if the group is circulant then the matrix of coefficients will be circulant and diagonalizable in the Fourier basis. If the group is symmetric and its elements are involutive, then the matrix will be diagonalizable in the Hadamard base. In these cases, information can be predominantly contained in the coefficient matrix but the implementation of the coefficient matrix, although inevitably costly in terms of gates, is much simpler than for any generic operator (see the article by Bullock and Markov for the implementation of diagonal operators [2]).

However among all the possible matrix groups, some are more suitable than others for a generic synthesis method. In the next section we narrow our research by investigating the theoretical properties of "good" groups for general synthesis.

3 Characterization of candidate groups G

In this section we discuss under which conditions the reuse method can be used as a generic method for the synthesis of circuits.

We can already eliminate the case where the group G is Abelian. Indeed, in this case the matrices of the group G commute in pairs and are therefore simultaneously diagonalizable, just like any member of the span of G. One cannot reach all unitary matrix but only those diagonalizable in a specific basis.

Ideally, the group G should be built easily for any number of qubits either with an adaptable construction for any n or with a recursive approach. In fact, we can show how to construct a solution group K and its matrix of coefficients for $n+m$ qubits from a solution group G for n qubits and a solution group H for m qubits.

We use can the properties of the tensor product to construct the group K. Indeed, by setting $K = G \otimes H$, provided that $U(2^n) \subseteq span(G)$ and $U(2^m) \subseteq span(H)$ then we have $U(2^{n+m}) \subseteq span(G \otimes H)$. Recall the identity

$$(g_1 \otimes g_2)(h_1 \otimes h_2) = (g_1 \otimes h_1)(g_2 \otimes h_2) \tag{5}$$

which is used to provide an expression of the coefficient matrix associated with K:

$$\begin{aligned} C_K = (\beta_{g^{-1}h})_{g,h\in G\otimes H} &= (\beta_{(g_1^{-1}\otimes h_1^{-1})^{-1}(g_2\otimes h_2)})_{g_1,g_2\in G,h_1,h_1\in H} \\ &= (\beta_{(g_1^{-1}g_2)\otimes(h_1^{-1}h_2)})_{g_1,g_2\in G,h_1,h_1\in H} \quad . \end{aligned} \tag{6}$$

With an appropriate ordering of K, the matrix C_K can be expressed as

$$C_K = \begin{pmatrix} \beta_{(g_1g_1,h_1h_1)} & \cdots & \beta_{(g_1g_n,h_1h_1)} & \beta_{(g_1g_1,h_1h_2)} & \cdots & \beta_{(g_1g_n,h_1h_2)} & \cdots \\ \vdots & & \vdots & \vdots & & \vdots & \\ \beta_{(g_ng_1,h_1h_1)} & \cdots & \beta_{(g_ng_n,h_1h_1)} & \beta_{(g_ng_1,h_1h_2)} & \cdots & \beta_{(g_ng_n,h_1h_2)} & \cdots \\ \beta_{(g_1g_1,h_2h_1)} & \cdots & \beta_{(g_1g_n,h_2h_1)} & \beta_{(g_1g_1,h_2h_2)} & \cdots & \beta_{(g_1g_n,h_2h_2)} & \cdots \\ \vdots & & \vdots & \vdots & & \vdots & \\ \beta_{(g_ng_1,h_2h_1)} & \cdots & \beta_{(g_ng_n,h_2h_1)} & \beta_{(g_ng_1,h_2h_2)} & \cdots & \beta_{(g_ng_n,h_2h_2)} & \cdots \\ \vdots & & \vdots & \vdots & & \vdots & \end{pmatrix}. \tag{7}$$

Thus, if a series of operations P factors C_G and a series of operations Q factors C_H, then $(P\otimes I)$ block-factorizes C_K and $(I\otimes Q)$ factorizes each block of C_K. Thus *a priori* $(P\otimes Q)$ factorizes C_K.

Therefore, if a solution for one qubit has been found, we can generate a solution for an arbitrary number of qubits by successive tensor products. Now, because the available memory is limited[1], it is desired to minimize the number of auxiliary qubits by logical qubits especially if additional qubits are necessary for error correcting codes [14]. In our study the size of the group G has been fixed to a maximum of 8 elements so as to have only 3 auxiliary qubits per logic bit. This accounts for the fact that quantum memory is expensive.

Only a few potential groups then satisfy the above restrictions:

- the projective Pauli group,
- the dihedral group over 3 qubits,
- the quaternion group.

4 Study of the candidates

The two 8-element groups – quaternion and dihedral group – are very similar: we only consider the latter. Indeed, the results on one of the two groups are immediately transposable to the other group.

This section analyzes first the base cases: the projective Pauli group and the dihedral group. In a second step, we discuss the behavior of the factorization mentioned in Section 3 for the case of two qubits in the context of the dihedral group.

[1] Simulating quantum computation on a conventional computer is known to be expensive [10] since a linear increase in the number of manipulated qubits yields an exponential increase in the size of the required memory.

4.1 The projective Pauli group

In the original publication, [7, Th. 6] has been extended to the case of projective groups in [7, Th. 7]. The particular projective group that we consider is

$$G = \{I, X, Z, XZ\}. \tag{8}$$

By setting $A = a_0 I + a_1 X + a_2 Z + a_3 XZ$ the associated coefficient matrix is

$$C_A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & -a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & -a_0 \end{pmatrix}. \tag{9}$$

Klappenecker and Rötteler [7, Eq. 12] gave the factorization

$$CNOT \times CNOT^{(2,1)} \times (H \otimes I_2) \times C_A \times (H \otimes I_2) \times CNOT = A \otimes I_2 \tag{10}$$

with

$$CNOT^{(2,1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

This shows that the synthesis of C_A is as difficult as the synthesis of A. No improvement can therefore be achieved with this group.

4.2 The dihedral group

The idea is to get rid of the projective character of the Pauli group by adding matrices to the G group, i.e with

$$G = \{I, -I, X, -X, Z, -Z, XZ, -XZ\}. \tag{11}$$

The coefficient matrix then becomes

$$C_A = \begin{pmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & a_7 \\ a_1 & a_0 & a_3 & a_2 & a_5 & a_4 & a_7 & a_6 \\ a_2 & a_3 & a_0 & a_1 & a_7 & a_6 & a_5 & a_4 \\ a_3 & a_2 & a_1 & a_0 & a_6 & a_7 & a_4 & a_5 \\ a_4 & a_5 & a_6 & a_7 & a_0 & a_1 & a_2 & a_3 \\ a_5 & a_4 & a_7 & a_6 & a_1 & a_0 & a_3 & a_2 \\ a_7 & a_6 & a_5 & a_4 & a_2 & a_3 & a_0 & a_1 \\ a_6 & a_7 & a_4 & a_5 & a_3 & a_2 & a_1 & a_0 \end{pmatrix}. \tag{12}$$

The best factorization that we have found is

$$P \times C_A \times P^\dagger = \begin{pmatrix} I & & & \\ & U & & \\ & & I & \\ & & & U \end{pmatrix} = I \otimes \Lambda(U) \tag{13}$$

where

$$P = (SWAP \otimes I) \times (I \otimes SWAP) \times (\Lambda(Z) \otimes I) \times H^{\otimes 3} \tag{14}$$

and where U is some arbitrary 2×2 unitary matrix, a priori not simpler to synthesize than the matrix A.

We can then conclude that no improvement can neither be found for this group.

4.3 Factorization for two qubits

In this section, we highlight the fact that the factorization procedure envisioned in Section 3 is not so simple to use, and that it does not necessarily provide a usable decomposition.

Consider indeed an operator A on 2 qubits. Using the dihedral group, the block factorization on C_A would then lead to

$$\begin{pmatrix} I & & & \\ & V & & \\ & & I & \\ & & & V \end{pmatrix} \tag{15}$$

with V a 2 by 2 block-matrix with blocks of size 8 by 8. Applying the same factorization on each block of V gives a matrix of the shape

$$\begin{pmatrix} I & & & & 0 & & & \\ & U_1 & & & & U_2 & & \\ & & I & & & & 0 & \\ & & & U_1 & & & & U_2 \\ 0 & & & & I & & & \\ & U_3 & & & & U_4 & & \\ & & 0 & & & & I & \\ & & & U_3 & & & & U_4 \end{pmatrix}, \tag{16}$$

with U_1, U_2, U_3 and U_4 arbitrary matrices of size 2×2, such that $\begin{pmatrix} U_1 & U_2 \\ U_3 & U_4 \end{pmatrix}$ is unitary. Synthesizing A therefore corresponds to synthesizing this matrix, which does not seem too less costly. This hints at the fact that extending the study to larger groups might not trivially help in getting a working solution.

5 Conclusion

We have recalled the fundamentals of the synthesis of quantum circuits. We started from an already existing method, aiming at implementing linear combinations of known circuits in order to attempt to derive a generic synthesis method. By clarifying how a generic synthesis method can be compositionally derived, we have illustrated the complexity of the problem. We presented the issues encountered when restricting the approach to small groups of one-qubit operators. This study calls for a more in-depth analysis of larger groups of two- or three-qubit operators.

References

1. Barenco, A., Bennett, C.H., Cleve, R., DiVincenzo, D.P., Margolus, N., Shor, P., Sleator, T., Smolin, J.A., Weinfurter, H.: Elementary gates for quantum computation. Physical Review A **52**(5), 3457 (1995)
2. Bullock, S.S., Markov, I.L.: Asymptotically optimal circuits for arbitrary n-qubit diagonal computations. Quantum Information and Computation **4**(1), 27–47 (2004)
3. Chi-Chih Yao, A.: Quantum circuit complexity. In: Proceedings of the 34th Annual Symposium on Foundations of Computer Science (SFCS'93), pp. 352–361. IEEE Computer Society, Washington, DC, USA (1993)
4. De Vos, A., De Baerdemacker, S.: Block-ZXZ synthesis of an arbitrary quantum circuit. Physical Review A **94**(5), 052,317 (2016)
5. Deutsch, D., Barenco, A., Ekert, A.: Universality in quantum computation. Proceedings of the Royal Society of London A **449**, 669–677 (1995)
6. JavadiAbhari, A., Patil, S., Kudrow, D., Heckey, J., Lvov, A., Chong, F.T., Martonosi, M.: ScaffCC: Scalable compilation and analysis of quantum programs. Parallel Computing **45**, 2–17 (2015)
7. Klappenecker, A., Rötteler, M.: Quantum software reusability. International Journal of Foundations of Computer Science **14**(05), 777–796 (2003)
8. Knill, E.: Approximation by quantum circuits. Tech. Rep. LANL report LAUR-95-2225, Los Alamos National Laboratory (1995)
9. Maslov, D., Dueck, G.W., Miller, D.M., Negrevergne, C.: Quantum circuit simplification and level compaction. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **27**(3), 436–444 (2008)
10. Nielsen, M.A., Chuang, I.L.: Quantum Computation and Quantum Information. Cambridge University Press (2011)
11. Reck, M., Zeilinger, A., Bernstein, H.J., Bertani, P.: Experimental realization of any discrete unitary operator. Physical Review Letters **73**(1), 58 (1994)
12. Saeedi, M., Arabzadeh, M., Zamani, M.S., Sedighi, M.: Block-based quantum-logic synthesis. Quantum Information and Computation **11**(3), 262–277 (2011)
13. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum-logic circuits. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25**(6), 1000–1010 (2006)
14. Steane, A.M.: Error correcting codes in quantum theory. Physical Review Letters **77**(5), 793 (1996)
15. Valiron, B., Ross, N.J., Selinger, P., Alexander, D.S., Smith, J.M.: Programming the quantum future. Communication of the ACM **58**(8), 52–61 (2015)
16. Vartiainen, J.J., Möttönen, M., Salomaa, M.M.: Efficient decomposition of quantum gates. Physical Review Letters **92**(17), 177,902 (2004)
17. Wecker, D., Svore, K.M.: LIQUi$|\rangle$: A software design architecture and domain-specific language for quantum computing (2014). ArXiv preprint 1402.4467.